Машинное обучение (Machine Learning)

Глубокие порождающие модели: вариационный автокодер (Deep Generative Learning: VAE)

Уткин Л.В.

Санкт-Петербургский политехнический университет Петра Великого

Содержание

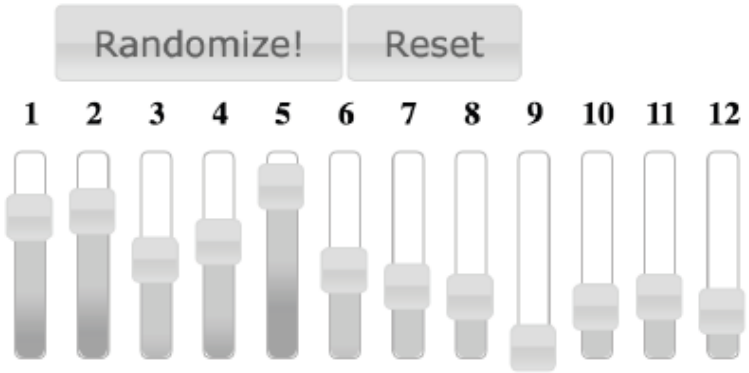
Порождающие модели

Мотивация

https://www.youtube.com/watch?v=XNZIN7Jh3Sg

Мотивация

http://www.dpkingma.com/sgvb_mnist_demo/demo.html



скрытые z

наблюдаемые x

Порождающие и разделяющие модели

1. **Разделяющие модели (дискриминативные)**: Оценка параметров на основе максимизации правдоподобия обучающей выборки по отношению к вероятности класса, а затем классификатор относит объект к его наиболее вероятному классу.
2. **Порождающие модели**: максимизация правдоподобия совместного распределения объектов и классов, а затем использование формулы Байеса для нахождения вероятности отношения объекта к классу.

Порождающие и разделяющие модели

- Пусть $(\mathbf{x}, y)$ - входные данные
- Результат обучения порождающей модели - совместное распределение вероятностей $p(\mathbf{x}, y)$
- Результат обучения разделяющей модели - условное распределение $p(y|\mathbf{x})$.
- Рассмотрим 4 точки данных:
  $(\mathbf{x}, y) \rightarrow \{(0,0),(0,0),(1,0),(1,1)\}$

$p(\mathbf{x}, y)$		
	$y=0$	$y=1$
$x=0$	$1/2$	0
$x=1$	$1/4$	$1/4$

$p(y\|\mathbf{x})$		
	$y=0$	$y=1$
$x=0$	1	0
$x=1$	$1/2$	$1/2$

Порождающие модели

Порождающая модель позволяет оценить совместное распределение вероятностей $p(\mathbf{x}, y)$. *Это означает, что можно сгенерировать* $\mathbf{x}, y$ *в соответствии с* $p(\mathbf{x}, y)$.

Порождающие и разделяющие модели

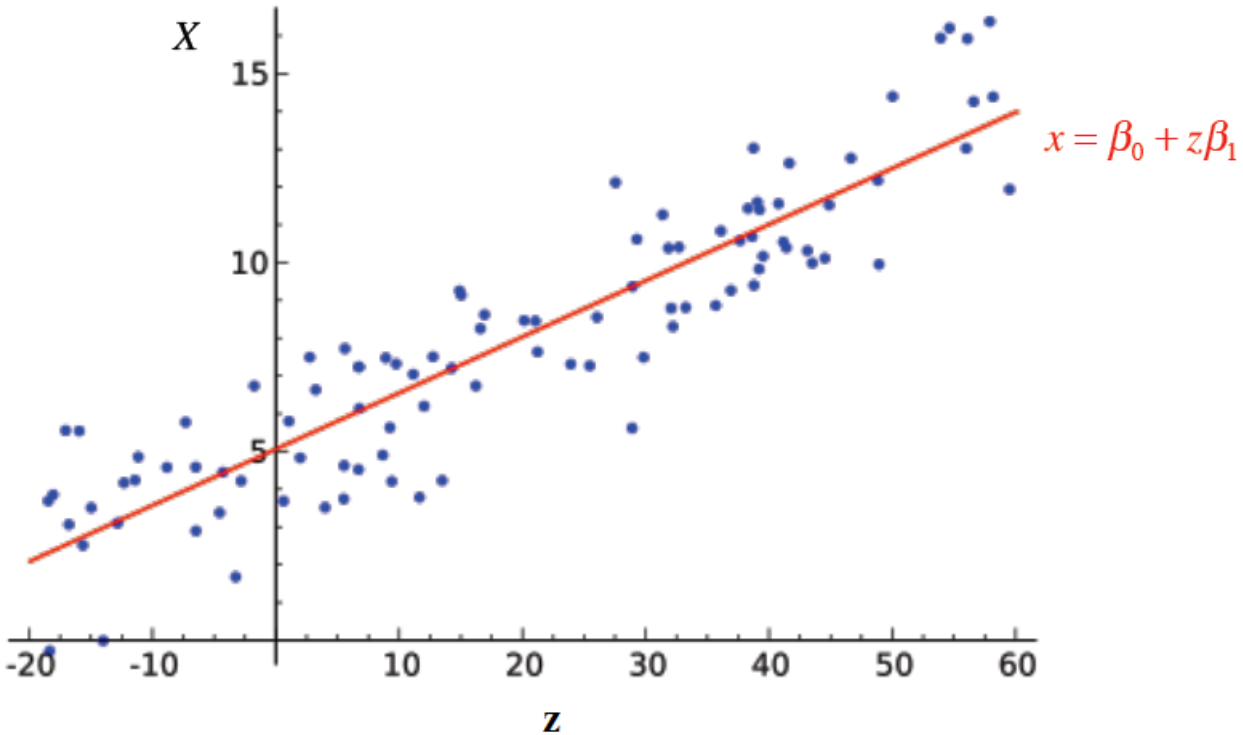
1. **Разделяющий** алгоритм пытается найти $p(y|\mathbf{x})$ из данных и затем классифицировать их или пытается найти разделяющую функцию между классами.
   Порождающий алгоритм пытается найти $p(\mathbf{x}, y)$, которое затем трансформируется в $p(y|\mathbf{x})$.
2. **Порождающий** алгоритм может использовать $p(\mathbf{x}, y)$, чтобы генерировать новые даные, аналогичные уже существующим.
   Разделяющий алгоритм в общем имеет лучше характеристики в задачах классификации.
3. **Порождающий**: Наивный Байес
   Разделяющий: Логистич. регрессия, SVM, Нейронные сети

Порождающие модели на основе нейронных сетей

1. **Вариационный автокодер (Variational AutoEncoder (VAE))**
2. Порождающие конкурирующие сети (Generative Adversarial Networks (GANs))
3. Глубокая машина Больцмана (DBM)
4. Глубокая сеть доверия (DBN)
5. Диффузионные модели

Вспомним регрессию

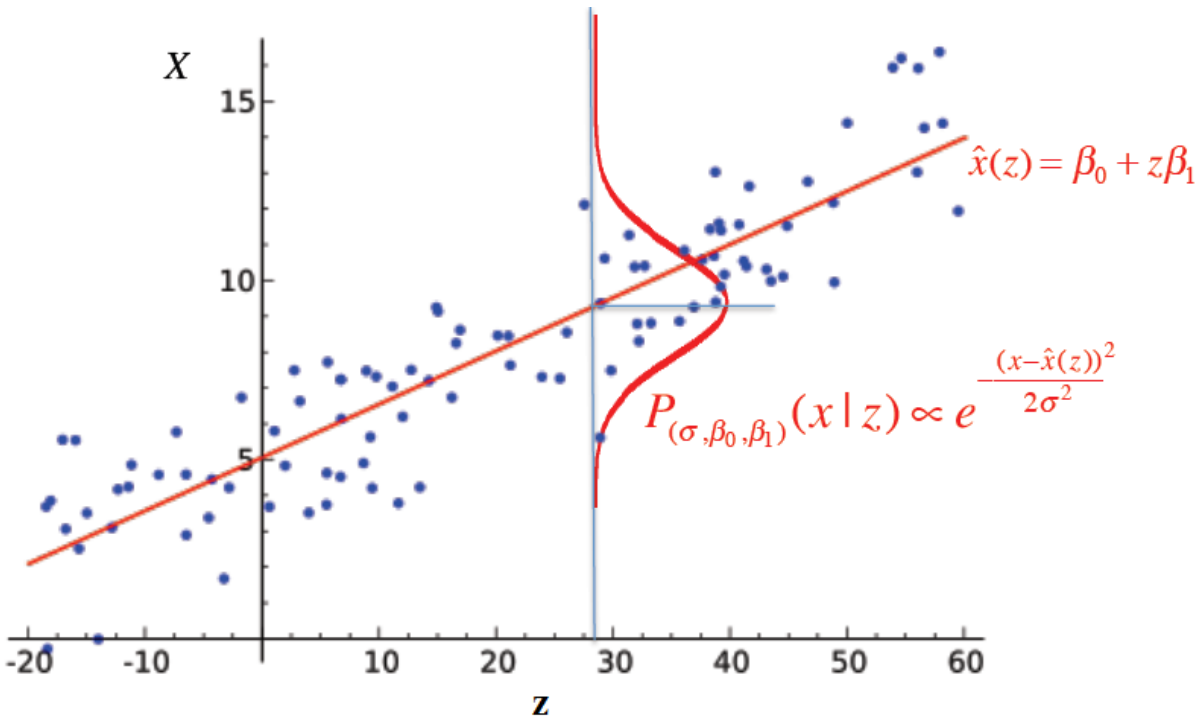
Многие представляют регрессию как множество точек и аппроксимирующую прямую



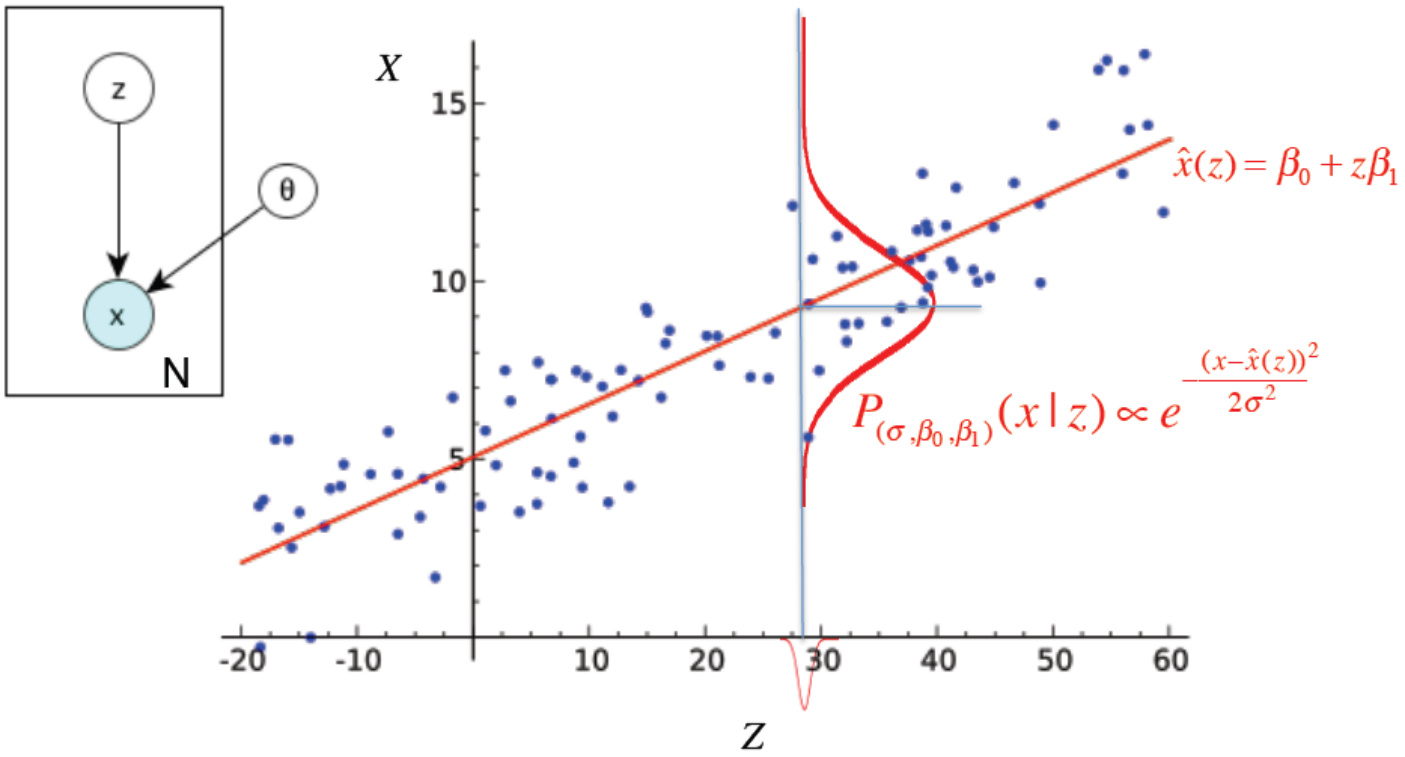
Вспомним регрессию еще

Статистики добавляют $P_\theta(X|Z)$: плюсы добавления модели ошибок:

- Какова вероятность точки данных
- Доверительные границы
- Сравниваемость моделей

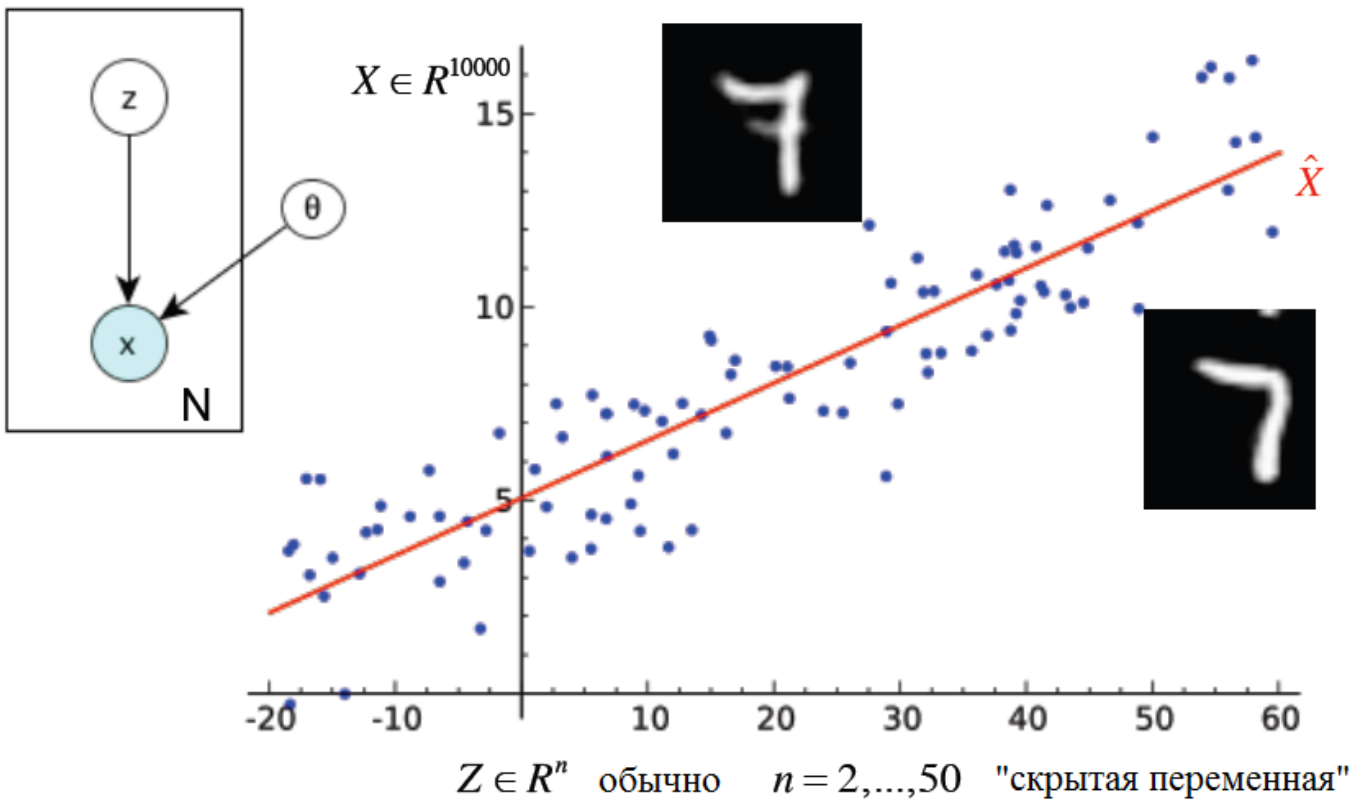


Вспомним регрессию с графическими моделями



Переход от малой размерности к большой

Переход от $\mathbb{R}^1$ к $\mathbb{R}^{10000}$



Скрытые переменные

- Рассмотрим генерацию изображений цифр от 0 до 9.
- Если левая половина цифры - левая половина 5, то правая половина не может быть левой половиной 0. Это помогает принимать решение какую цифру генерировать.
- Такое решение отражается в скрытой переменной z (latent variable).
- Перед тем, как модель нарисует что-нибудь, она сначала сгенерирует число z из $\{0, ..., 9\}$, затем убеждается, все ли штрихи совпадают с этой цифрой.
- z называется “скрытой”, так как при данной цифре, порожденной моделью, мы не знаем точно, какие параметры z сгенерировали цифру.

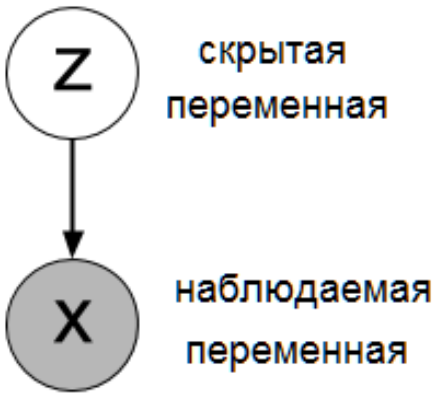
Скрытые переменные

Неформально: Чтобы говорить о качестве модели, необходимо удостовериться, что для каждой точки $\mathbf{x}$ в обучающей выборке S, есть "окружение" скрытой переменной, которое заставляет модель генерировать что-то очень похожее на $\mathbf{x}$.

Формально:

- Дано: вектор скрытых переменных $\mathbf{z} \sim p(\mathbf{z})$; семейство функций $f(\mathbf{z}; \theta)$, θ - вектор параметров
- Нужно оптимизировать θ так, что $f(\mathbf{z}; \theta)$ производит выборку подобно $\mathbf{x}$ с высокой вероятностью для каждого $\mathbf{x} \in S$, когда $\mathbf{z}$ генерируется из $p(\mathbf{z})$

Некоторое отступление



Пусть **x** представляет “исходные значения пикселей изображения”, а z - двоичная переменная такая, что $z = 1$, если “**x** - изображение кота”.

Некоторое отступление

$X=$ 

$P(z)=1$ (точно кот)

$X=$ 

$P(z)=0$ (точно не кот)

$X=$ 

$P(z)=0.1$ (напоминает кота)

Некоторое отступление

- Теорема Байеса устанавливает связь между переменными:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$$

- $p(\mathbf{z}|\mathbf{x})$ - апостериорная вероятность: “дано изображение, какова вероятность, что это кот?” Если можно сгенерировать $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})$, то значения можно использовать для построения классификатора, который скажет, является ли данное изображение котом или нет.

Некоторое отступление

- $p(\mathbf{x}|\mathbf{z})$ - правдоподобие: “дано значение $\mathbf{z}$, как вероятно, что изображение $\mathbf{x}$ определенной категории $\{$ кот / не кот$\}$ ”.
- Если можно сгенерировать $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$, то можно сгенерировать изображения котов и изображения не котов также просто, как генерацию случайных чисел.
- $p(\mathbf{z})$ - априорная веротяность (любая информация о $\mathbf{z}$), например, если известно, что $1/3$ всех изображений - коты, то $p(\mathbf{z} = 1) = 1/3$ и $p(\mathbf{z} = 0) = 2/3$.

А это важно - здесь суть подхода

Скрытые переменные можно интерпретировать в рамках байесовского подхода как априорные доверия, связанные с наблюдаемыми переменными.

Например, если мы полагаем, что $\mathbf{x}$ *имеет многомерное нормальное распределение, то скрытая переменная* $\mathbf{z}$ *может представлять среднее и дисперсию нормального распределения. Распределение* $p(\mathbf{z})$*, определенное на параметрах, является априорным для* $p(\mathbf{x})$*.*

Генерируя $\mathbf{z}$ *(различные параметры), мы можем генерировать различные* $\mathbf{x}$*!*

В чем проблема?

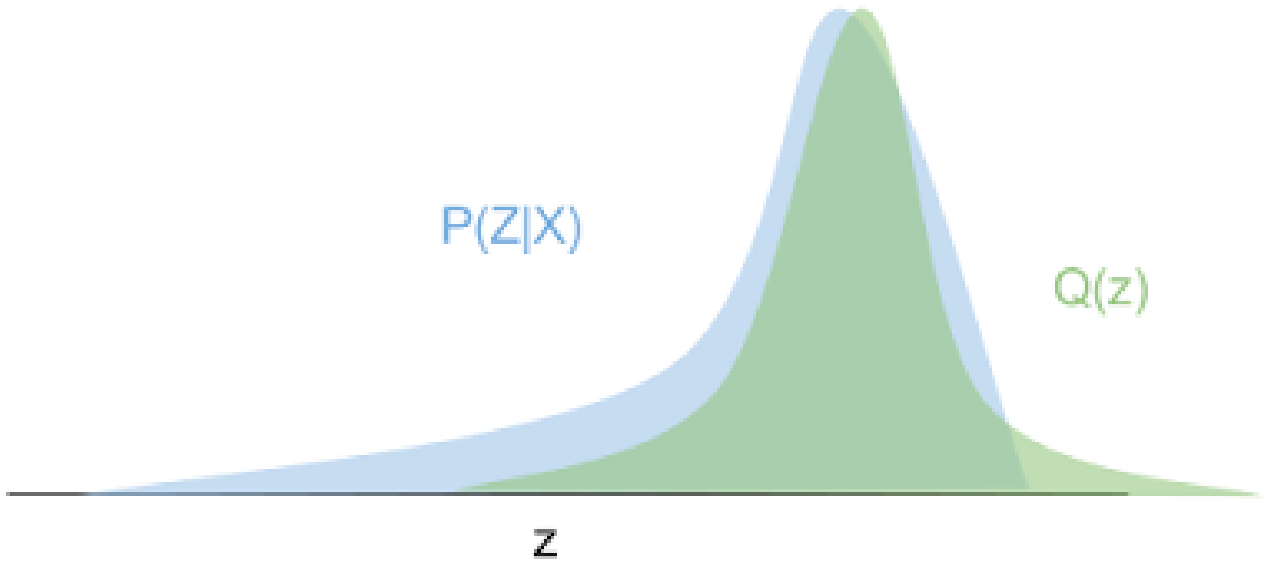
- Для сложных задач мы не знаем, как генерировать из $p(\mathbf{z}|\mathbf{x})$ или как вычислить $p(\mathbf{x}|\mathbf{z})$.
- Мы знаем форму $p(\mathbf{z}|\mathbf{x})$, но соответствующее вычисление настолько является сложным, что мы не можем оценить его за приемлемое время.
- Можно попытаться использовать какой-нибудь известный подход для этого, но большинство из них медленно сходится.

Идея вариационного вывода

Вместо сложного распределения $p(\mathbf{z}|\mathbf{x})$, используем простое параметрическое распределение $q_\phi(\mathbf{z}|\mathbf{x})$, например, нормальное, для которого известно, как получить апостериорное распределение. При этом мы подбираем параметры ϕ таким образом, чтобы q_ϕ было как можно ближе к $p(\mathbf{z}|\mathbf{x})$.

Близость определяется, например, при помощи дивергенции Кульбака-Лейблера.

Иллюстрация близких распределений



Совсем формально

$$p_\theta(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z};\theta)p_\theta(\mathbf{z})d\mathbf{z} = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}p(\mathbf{x}|\mathbf{z};\theta) \to \max_\theta$$

- $f(\mathbf{z};\theta)$ заменена распределением $p(\mathbf{x}|\mathbf{z};\theta)$
- Если модель способна воспроизвести обучающие примеры, то она также способна с большой вероятностью породить аналогичные и с малой вероятностью неподобные примеры.

Забегая вперед

- В вариационном автокодере, если $\mathbf{x} \in \mathbb{R}$, то $p(\mathbf{x}|\mathbf{z};\theta) = \mathcal{N}(\mathbf{x}|f(\mathbf{z};\theta), \sigma^2 \cdot I)$ - нормальное распределение со средним $f(\mathbf{z};\theta)$ и дисперсией $\sigma^2 \cdot I$
- Тогда можно увеличивать постепенно $p(\mathbf{x}|\mathbf{z};\theta)$ делая $f(\mathbf{z};\theta)$ близким к $\mathbf{x}$ для некоторого $\mathbf{z}$
- Если $\mathbf{x}$ - двоичное, то then $p(\mathbf{x}|\mathbf{z};\theta)$ - распределение Бернулли с параметром $f(\mathbf{z};\theta)$

Забегая еще далее

Для решения задачи с $p(\mathbf{x})$ имеются 2 проблемы:

1. как определить скрытые переменные $\mathbf{z}$, т.е. решить, какую информацию они представляют?
2. что делать с интегрированием по $\mathbf{z}$?

Вариационный автокодер позволяет ответить на эти вопросы

Как определить скрытые переменные

Мы хотим в идеале избежать:

- принятия решения “в ручную”, какую информацию каждая координата вектора $\mathbf{z}$ кодирует
- в явном виде описывать зависимости (скрытую структуру) между координатами $\mathbf{z}$

Вариационный автокодер (VAE)

Подход VAE

- Вместо отображения входных данных в фиксированный вектор мы хотим отобразить их в распределение p_θ с параметрами θ.
- Связь между входными данными $\mathbf{x}$ и скрытым вектором кодирования $\mathbf{z}$ может быть полностью определена следующим образом:
  - Априорное: $p_\theta(\mathbf{z})$
  - Правдоподобие: $p_\theta(\mathbf{x}|\mathbf{z})$
  - Апостериорное: $p_\theta(\mathbf{z}|\mathbf{x})$

Подход VAE

- Предполагая, что мы знаем реальный параметр θ^* для этого распределения.
- Чтобы генерировать пример, который выглядит как реальная точка данных $\mathbf{x}^{(i)}$, мы выполняем следующие шаги:
  1. Сначала берем пример $\mathbf{z}$ из априорного распределения $p_{\theta^*}(\mathbf{z})$.
  2. Затем вектор $\mathbf{x}^{(i)}$ генерируется из условного распределения $p_{\theta^*}(\mathbf{x}|\mathbf{z} = \mathbf{z}^{(i)})$.

Подход VAE

- Оптимальный параметр θ^* максимизирует вероятность генерации реальных выборок данных:

$$\theta^* = \arg\max_\theta \prod_{i=1}^{n} p_\theta(\mathbf{x}^{(i)}).$$

- Или то же самое:

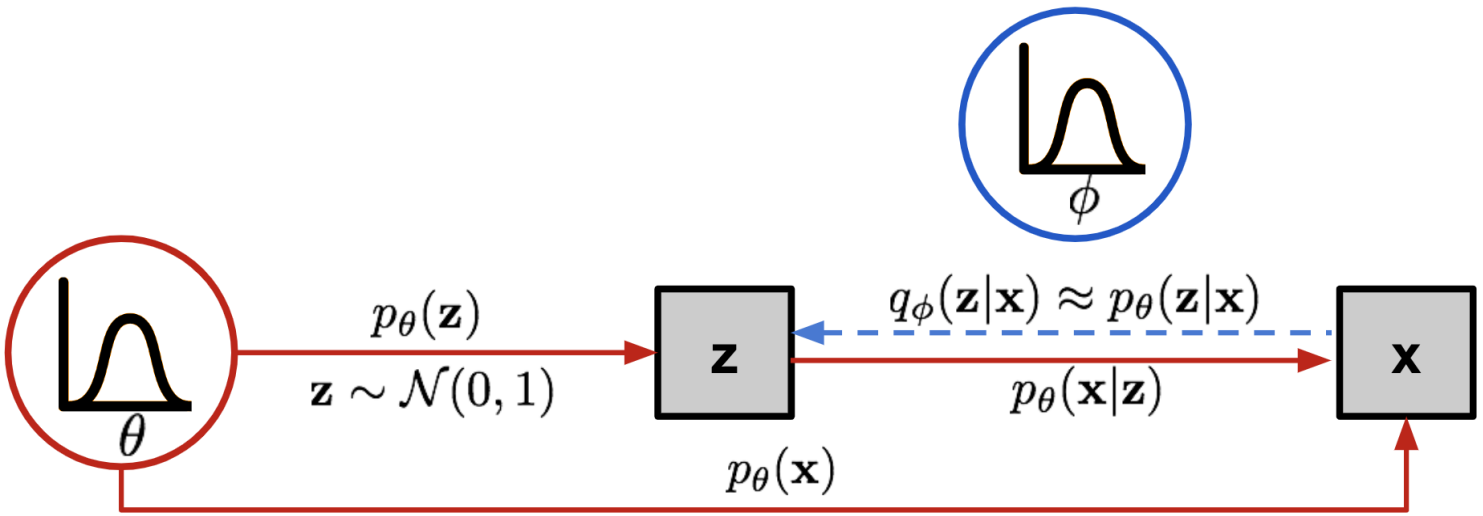
$$\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(\mathbf{x}^{(i)}).$$

- Обновим уравнение, чтобы лучше увидеть процесс генерации данных и включить вектор кодирования:

$$p_\theta(\mathbf{x}^{(i)}) = \int p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) \cdot p_\theta(\mathbf{z}) d\mathbf{z}.$$

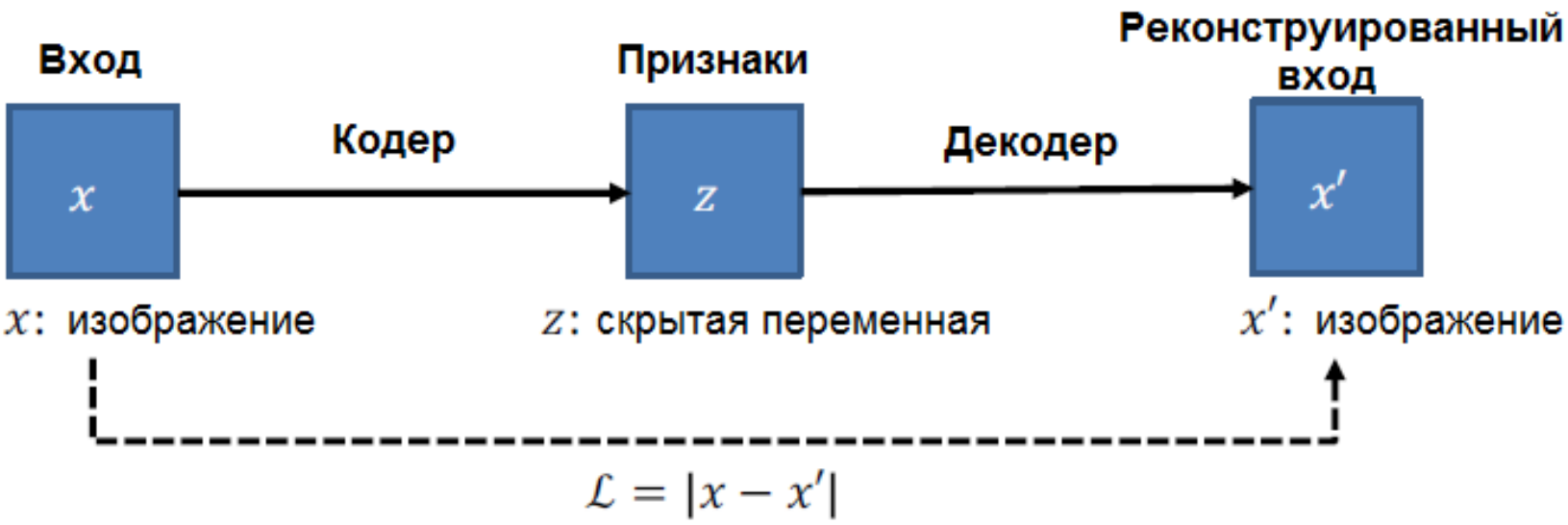
Подход VAE

- $p_\theta(\mathbf{x}^{(i)}) = \int p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) \cdot p_\theta(\mathbf{z})d\mathbf{z}$
- К сожалению, вычислить $p_\theta(\mathbf{x}^{(i)})$ непросто. Поэтому введем новое приближение $q_\phi(\mathbf{z}|\mathbf{x})$ с параметрами ϕ.

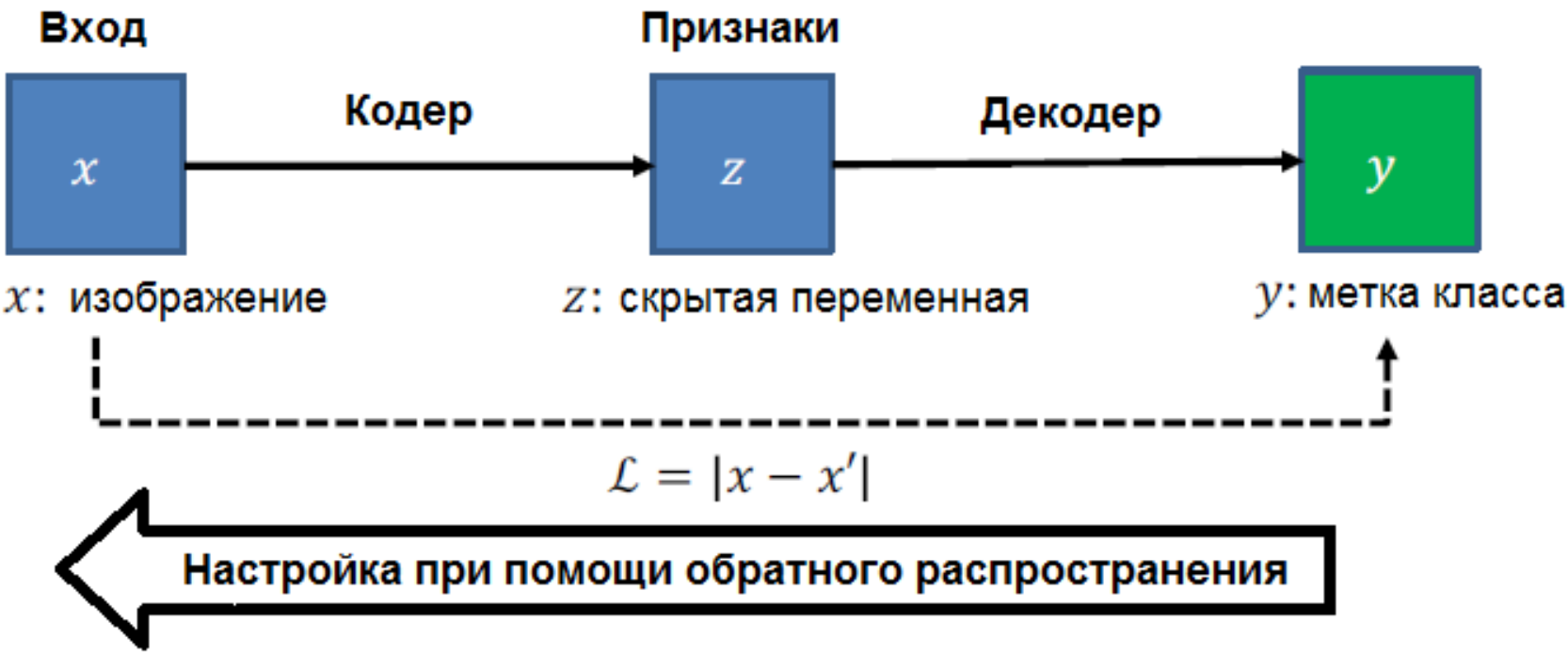


- Условная вероятность $p_\theta(\mathbf{x}|\mathbf{z})$ определяет генеративную модель (вероятностный декодер).
- Аппроксимация $q_\phi(\mathbf{z}|\mathbf{x})$ является вероятностным кодировщиком.

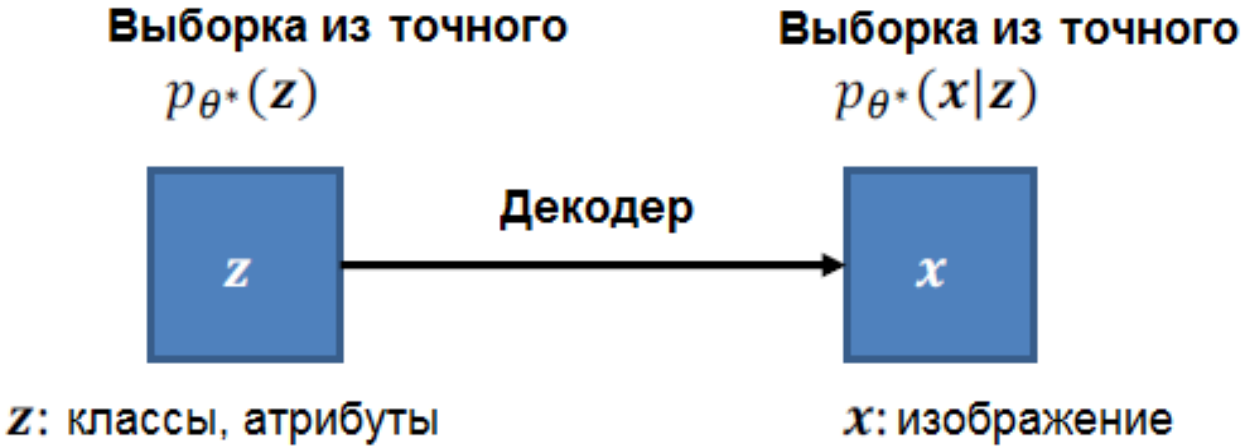
Стандартный автокодер для представления признаков



Стандартный автокодер для классификации



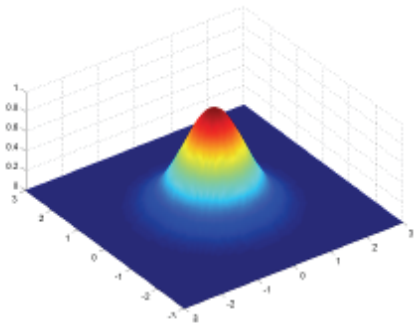
Вариационный автокодер

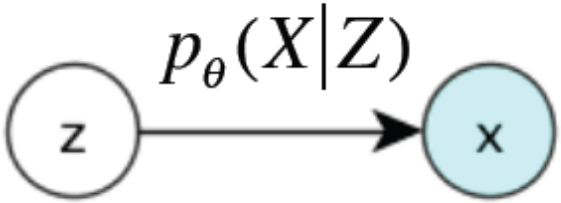


- Значение генерируется из априорного распределения $p_{\theta*}(\mathbf{z})$
- Значение $\mathbf{x}^{(i)}$ генерируется из условного распределения $p_{\theta*}(\mathbf{x}|\mathbf{z})$
- Точное значение параметра θ^* и значения скрытых переменных $\mathbf{z}^{(i)}$ не известны

Вариационный автокодер (кодирование)

Дано: $\mathbf{z} \sim \mathcal{N}(0, I)$, $\mathbf{x}|\mathbf{z} \sim p_{\theta*}(\mathbf{x}|\mathbf{z})$

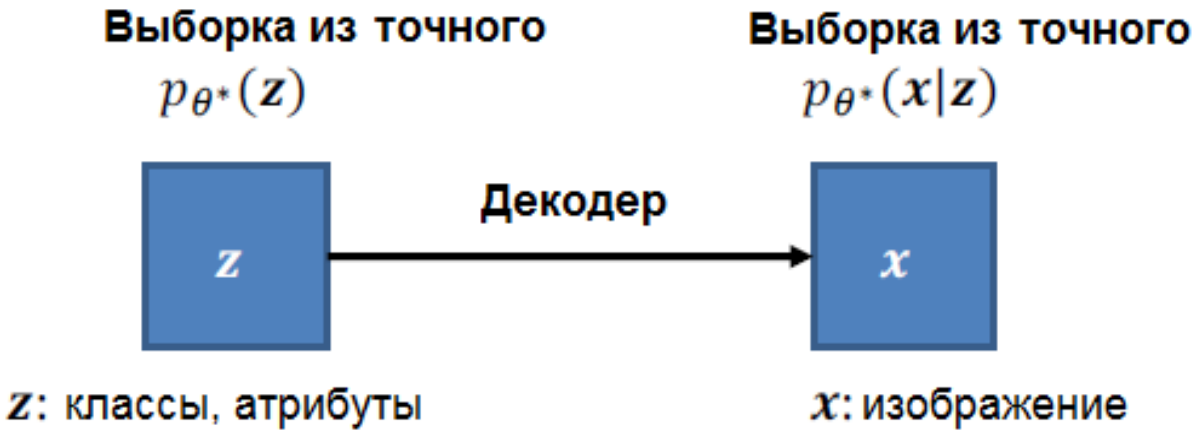




Один пример

Хотим получить (обучиться) θ из N обучающих наблюдений $\mathbf{x}^{(i)}$, $i = 1, ..., N$

Вариационный автокодер



- Трудно оценить $p_{\theta^*}(\mathbf{z})$ и $p_{\theta^*}(\mathbf{x}|\mathbf{z})$ на практике
- Необходимо аппроксимировать эти распределения

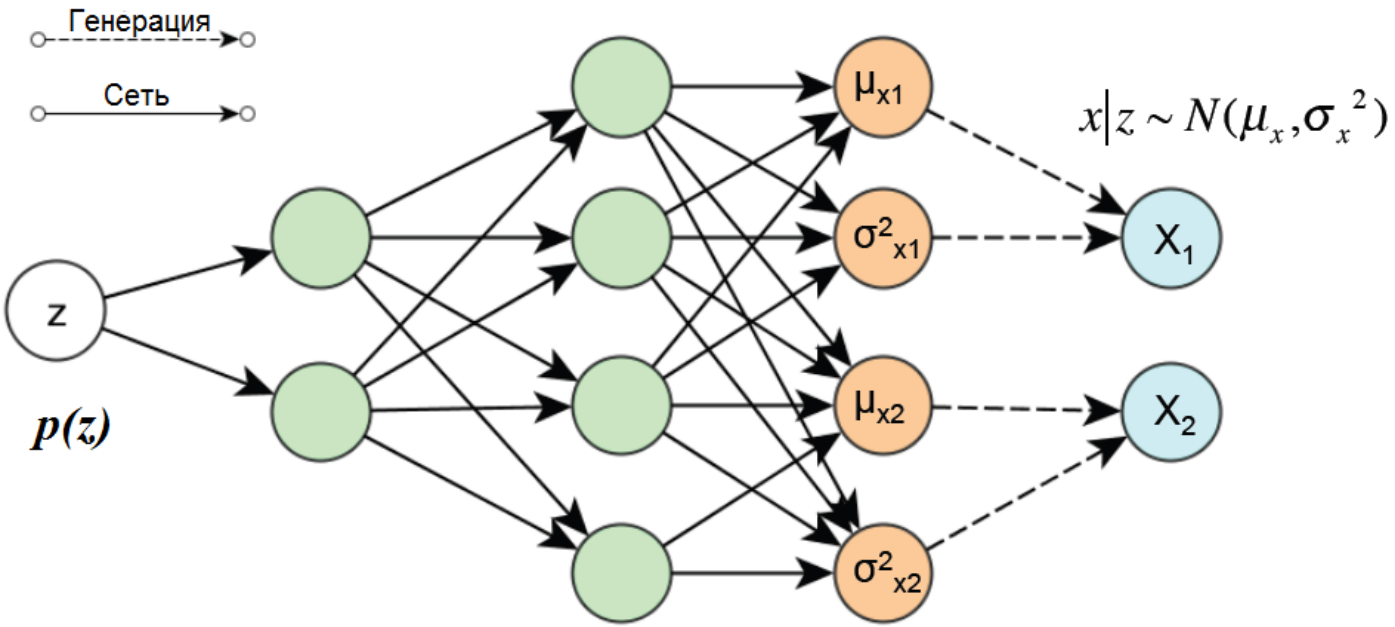
Вариационный автокодер - основная идея

- Предполагаем, что $p_\theta(\mathbf{z})$ имеет нормальное распределение
- Предполагаем, что $p_\theta(\mathbf{x}|\mathbf{z})$ имеет диагональное нормальное распределение
- Декодер оценивает среднее и дисперсию $p_\theta(\mathbf{x}|\mathbf{z})$

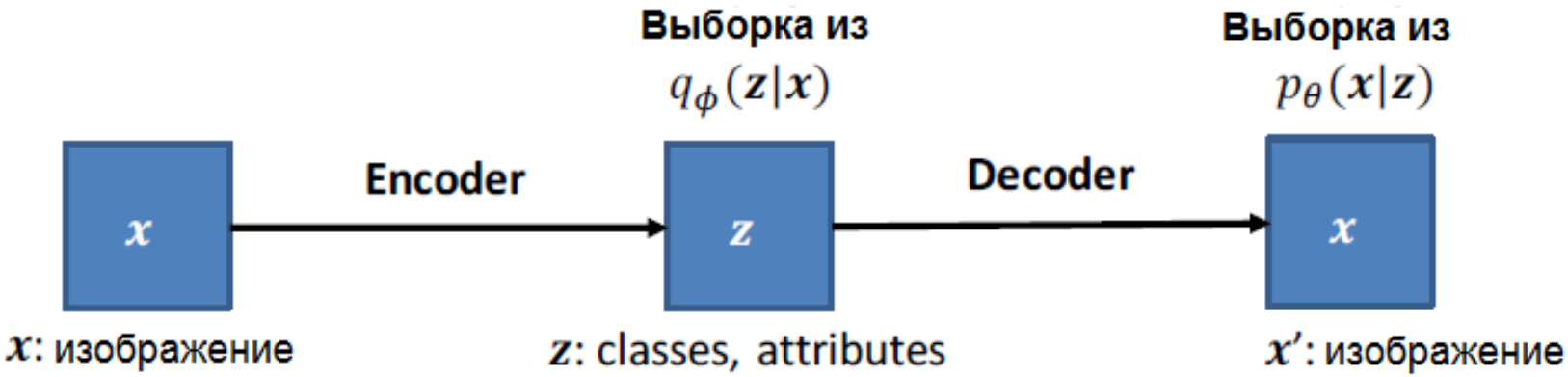
Декодирование

Пусть $z \in \mathbb{R}$ и $x \in \mathbb{R}^2$

Идея: Нейр.сеть + Норм. распр (или Бернулли) с диагональной ковариацией Σ



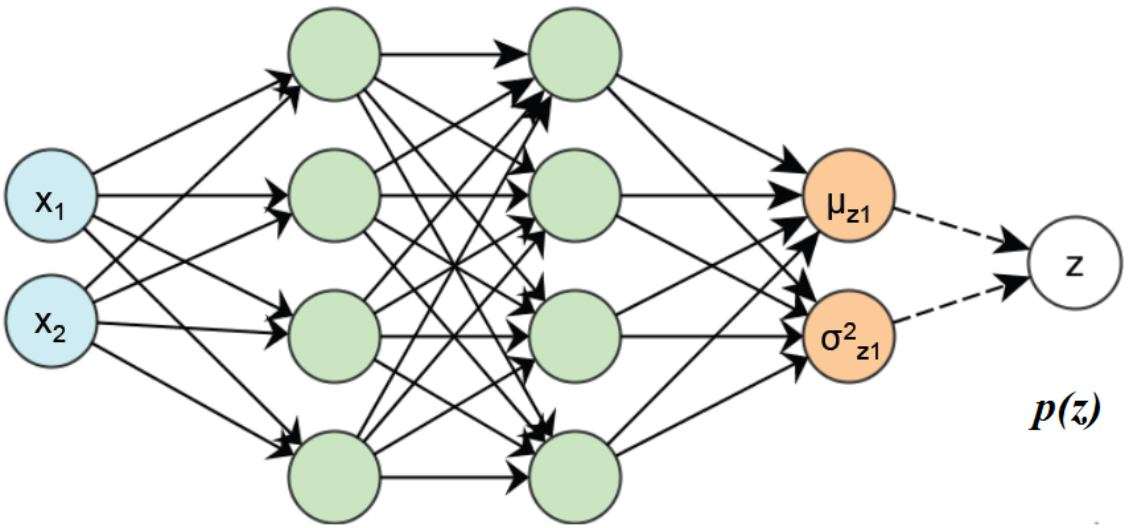
Вариационный автокодер - основная идея



- Кодер оценивает среднее и дисперсию $q_\phi(\mathbf{x}|\mathbf{z})$
- Декодер оценивает среднее и дисперсию $p_\theta(\mathbf{x}|\mathbf{z})$
- Максимизируем нижнюю границу маргинального правдоподобия $p_\theta(\mathbf{x}|\mathbf{z})$

Декодирование

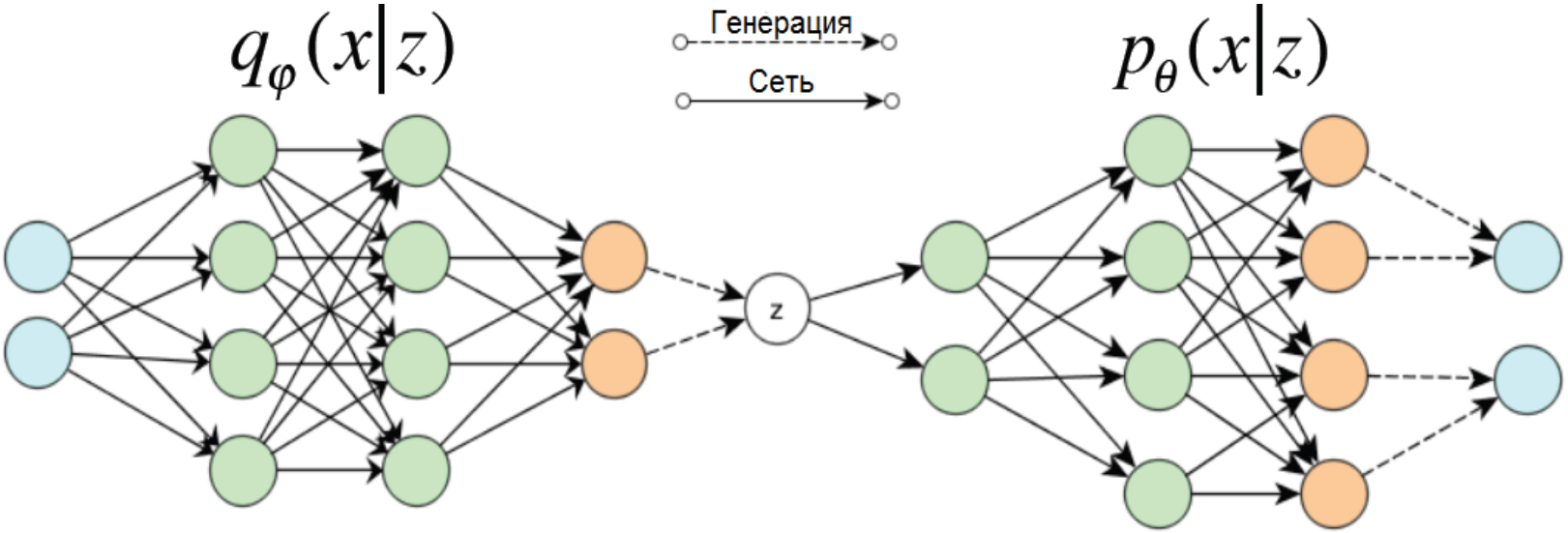
NN прямого распространения + Gaussian:
$q_\phi(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mu_\mathbf{z}(\mathbf{x}), \sigma_\mathbf{z}(\mathbf{x}))$



Таким образом

Если генерация $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$ является “кодированием”, которое конвертирует наблюдение $\mathbf{x}$ в скрытый код $\mathbf{z}$, то генерация $\mathbf{x} \sim q(\mathbf{x}|\mathbf{z})$ - “декодирование”, которое реконструирует наблюдения из $\mathbf{z}$.

Весь автокодер



- Параметры φ и θ обучаются при помощи обратного распространения
- Главное определить функцию потерь

Выбор функции потерь: ELBO (evidence lower bound)

- Какая техника в статистике является одной из лучших?
- Метод максимального правдоподобия: настройка ϕ и θ, чтобы максимизировать функцию правдоподобия
- Максимизируем логарифм правдоподобия для заданного "изображения" $\mathbf{x}^{(i)}$ обучающего множества
- Суммируем по всем обучающим примерам

Выбор функции потерь: ELBO

- Оценка апостериорного распределения $q_\phi(\mathbf{z}|\mathbf{x})$ должна быть близка к реальному $p_\theta(\mathbf{z}|\mathbf{x})$.
- Дивергенция Кульбака-Лейблера $KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$ для оценки расстояния между этими двумя распределениями.
- $KL(X||Y)$ оценивает, сколько информации теряется, если распределение Y используется для представления X.
- В нашем случае мы хотим минимизировать $KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$ по ϕ.

Нижняя граница функции правдоподобия

$$L = \log p(\mathbf{x}) = \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}) = \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \left(\frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right) =$$

$$= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \left(\frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right) =$$

$$= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \left(\frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \right) + \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \left(\frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right) =$$

$$= L^v + KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) \geq L^v$$

- KL определяет, насколько $q(\mathbf{z}|\mathbf{x})$ близко к $p(\mathbf{z}|\mathbf{x})$
- L^v - **нижняя граница** для правдоподобия; $L^v = L$ при $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x})$

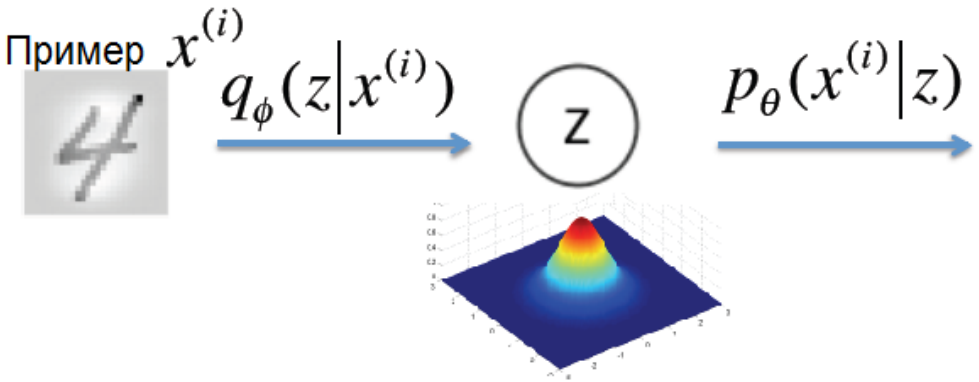
Приближенный вывод (1)

$$
\begin{aligned}
L^v &= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \left(\frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \right) = \\
&= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \left(\frac{p(\mathbf{x}|\mathbf{z}) \cdot p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right) = \\
&= \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \left(\frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right) + \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \left(p(\mathbf{x}|\mathbf{z}) \right) = \\
&= -KL(q(\mathbf{z}|\mathbf{x}^{(i)})||p(\mathbf{z})) + \mathbb{E}_{q(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p(\mathbf{x}^{(i)}|\mathbf{z}) \right]
\end{aligned}
$$

Приближенный вывод (2)

$$L^v = -KL(q(\mathbf{z}|\mathbf{x}^{(i)})||p(\mathbf{z})) + \mathbb{E}_{q(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p(\mathbf{x}^{(i)}|\mathbf{z})\right]$$

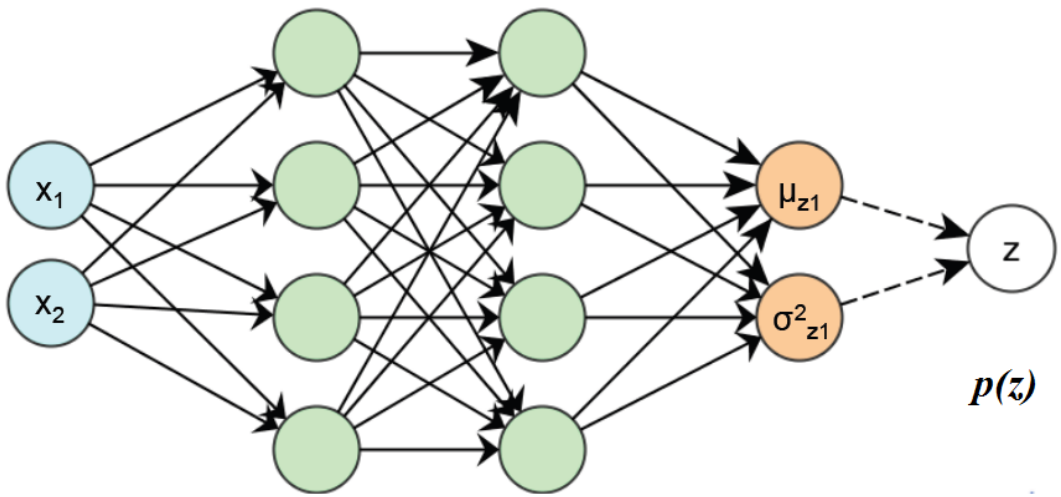
- $-KL(q(\mathbf{z}|\mathbf{x}^{(i)})||p(\mathbf{z}))$ - **регуляризация** $p(\mathbf{z})$ - обычно $\mathcal{N}(\mathbf{z}; 0, 1)$
- $\mathbb{E}_{q(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p(\mathbf{x}^{(i)}|\mathbf{z})\right]$ - **качество реконструкции**, $\log(1)$, если $\mathbf{x}^{(i)}$ реконструируется идеально ($\mathbf{z}$ образует $\mathbf{x}^{(i)}$)



Вычисление регуляризациии

- Используем $\mathcal{N}(\mathbf{z}; 0, 1)$ как априорное для $p(\mathbf{z})$
- $q(\mathbf{z}|\mathbf{x}^{(i)})$ - норм. с параметрами $\mu_{\mathbf{z}}^{(i)}, \sigma_{\mathbf{z}}^{(i)}$, определяемыми NN

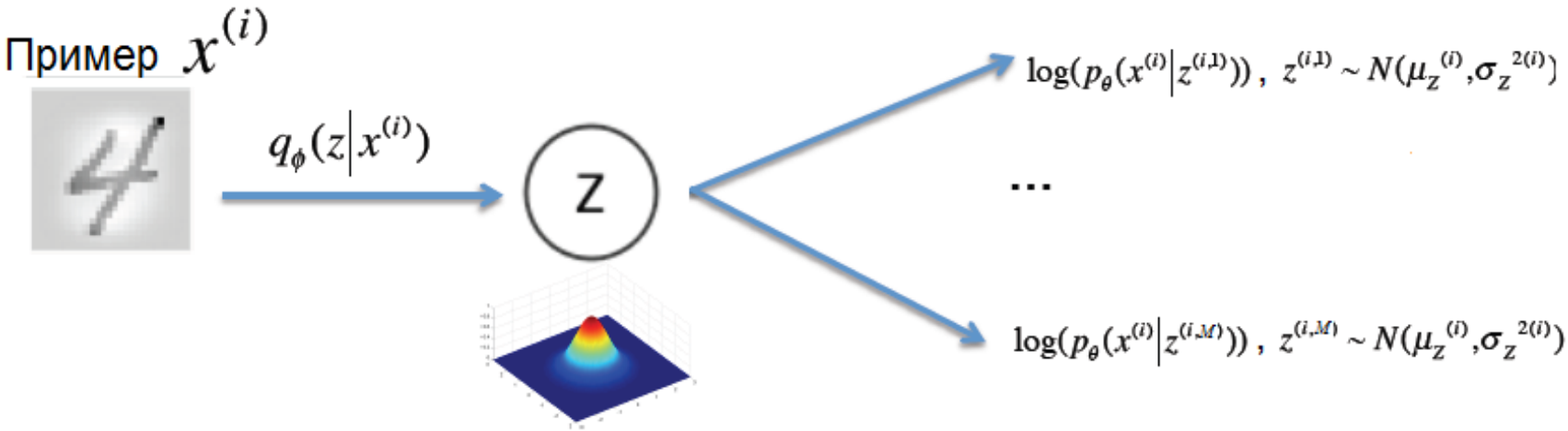
$$-KL(q(\mathbf{z}|\mathbf{x}^{(i)})||p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log\left(\sigma_{\mathbf{z}_j}^{(i)2}\right) - \mu_{\mathbf{z}_j}^{(i)2} - \sigma_{\mathbf{z}_j}^{(i)2}\right)$$



Вычисление качества реконструкции

Приближенное вычисление $\mathbb{E}_{q(\mathbf{z}|\mathbf{x}^{(i)})}$ генерацией $\mathbf{z}^{(i,l)} \sim q(\mathbf{z}|\mathbf{x}^{(i)})$, $l = 1, ..., M$:

$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p(\mathbf{x}^{(i)}|\mathbf{z})\right] \approx \frac{1}{M}\sum_{i=1}^{M} \log\left(p(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})\right)$$



Репараметризация

- Обратное распространение невозможно при случайной генерации
- Используется прием репараметризации

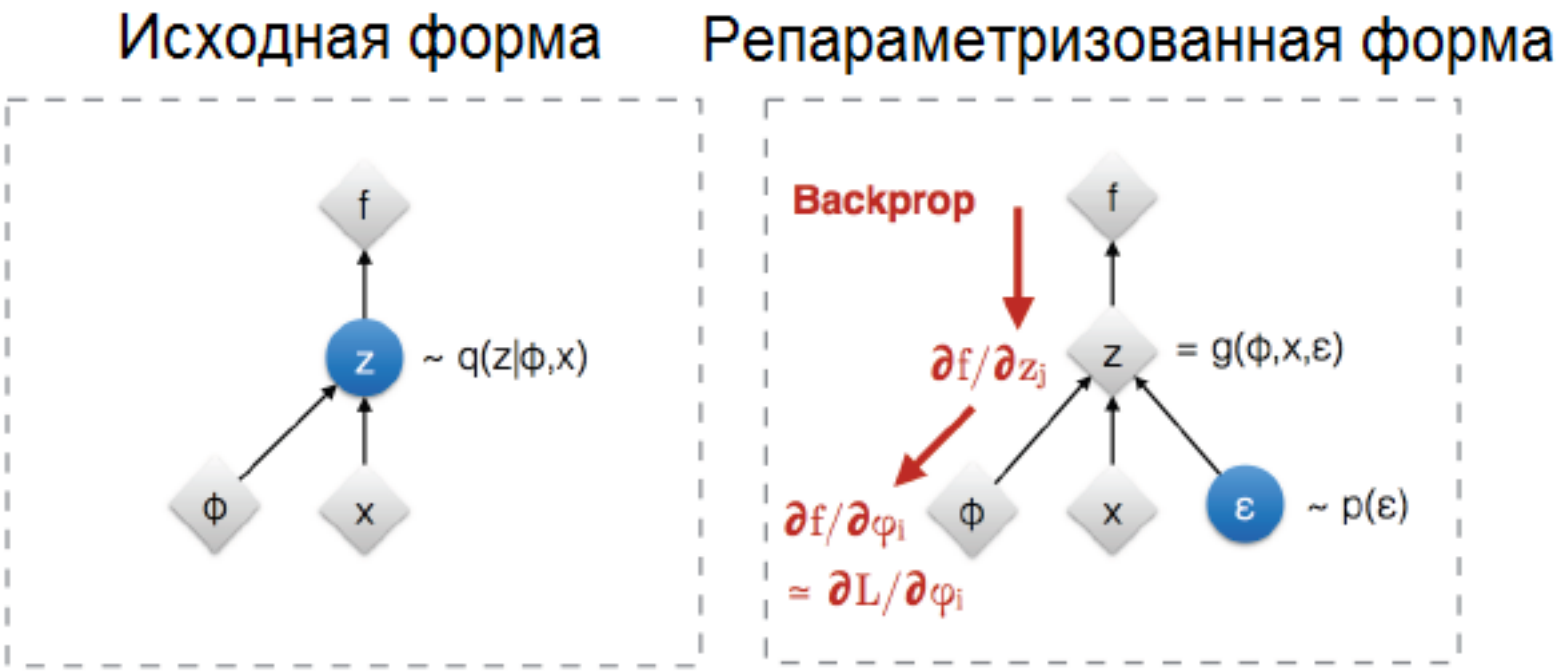
$$\mathbf{z}^{(i,l)} \sim \mathcal{N}\left(\mu_{\mathbf{z}}^{(i)}, \sigma_{\mathbf{z}}^{(i)}\right) \rightarrow \mathbf{z}^{(i,l)} = \mu_{\mathbf{z}}^{(i)} + \sigma_{\mathbf{z}}^{(i)} \odot \varepsilon_i,\ \varepsilon_i \sim \mathcal{N}(0, 1)$$

или

$$\mathbf{z} = \mu_{\mathbf{z}} + \sigma_{\mathbf{z}} \odot \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \mathbf{I})$$

- $\mathbf{z}$ имеет то же распределение, но теперь возможно обратное распространение, т.к. есть детерминированная часть и шум

Иллюстрация репараметризации

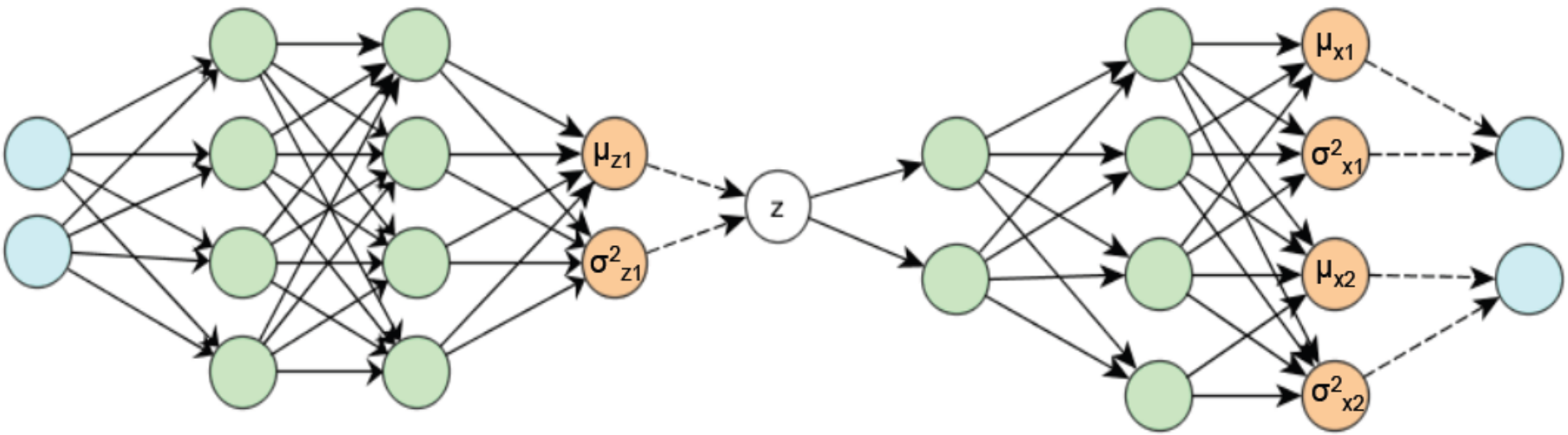


: Детерминированная вершина

: Случайная вершина

[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Объединяем все вместе (1)



Объединяем все вместе (2)

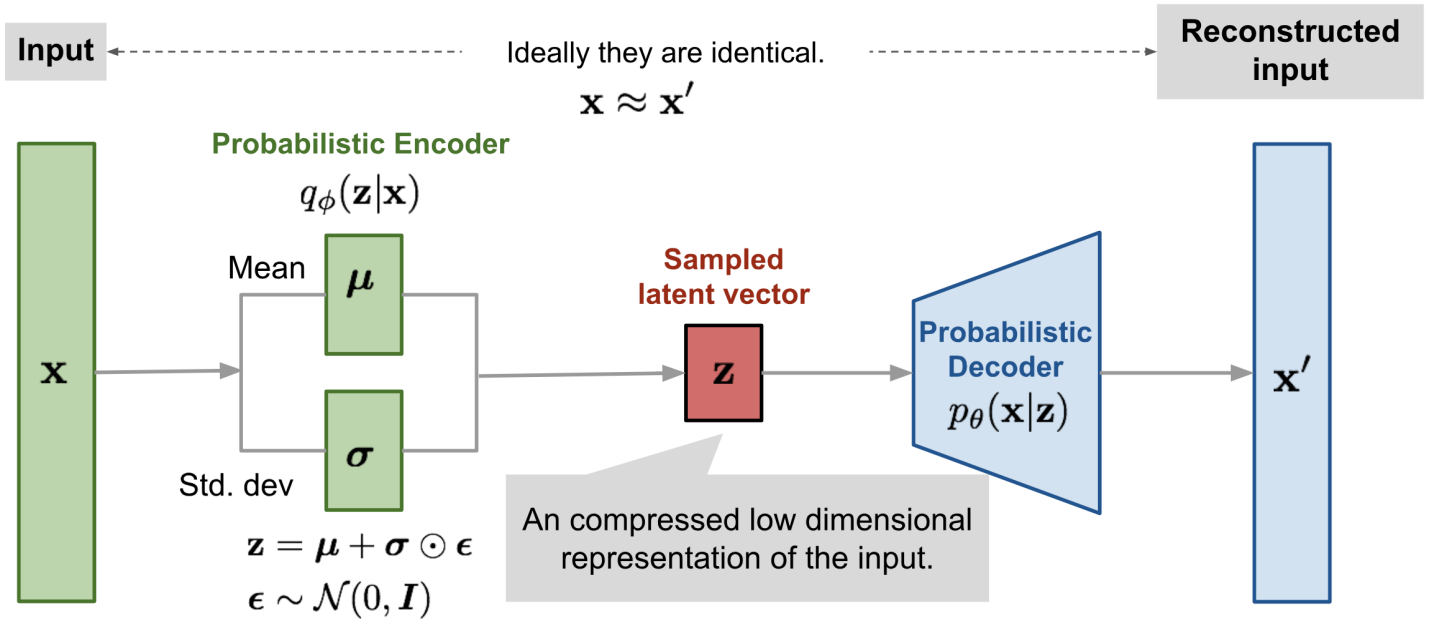
- **Регуляризация**: град. спуск, чтобы оптимизировать по $\mathbf{x}^{(i)}$

$$-KL(q(\mathbf{z}|\mathbf{x}^{(i)})||p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log\left(\sigma_{\mathbf{z}_j}^{(i)2}\right) - \mu_{\mathbf{z}_j}^{(i)2} - \sigma_{\mathbf{z}_j}^{(i)2}\right)$$

- **Репродукция**: метод наим. квад. для постоянной дисперсии

$$-\log p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)}) = \sum_{i=1}^{M}\frac{1}{2}\log\left(\sigma_{\mathbf{x}_i}^2\right) + \frac{\left(\mathbf{x}_j^{(i)} - \mu_{\mathbf{x}_j}\right)^2}{2\sigma_{\mathbf{x}_i}^2}$$

Объединяем все вместе (3)



https://lilianweng.github.io/posts/2018-08-12-vae/

Вариационные методы и Deep Learning

- Deep learning - эффективный инструментарий для оптимизации, например, методом градиентного спуска, когда имеется огромное количество параметров и данных.
- Вариационные байесовские методы являются аппаратом, при помощи которого можно “переписать” задачи статистического вывода в виде задач оптимизации.
- Комбинация вариационных байесовских методов и Deep learning позволяет реализовать вывод на очень сложных апостериорных распределениях вероятностей.

Beta-VAE (1)

- Если каждая переменная в скрытом представлении чувствительна только к одному единственному порождающему фактору и относительно инвариантна к другим факторам, будем говорить, что это представление распутано (disentangled) или факторизовано.
- Одно из преимуществ распутанного представления - хорошая интерпретируемость и простота обобщения.
- Например, модель, обученная на фотографиях чел-ких лиц, может фиксировать цвет кожи, цвет волос, длину волос, эмоции, наличие очков и многие другие относительно независимые факторы в отдельных измерениях. Такое распутанное представление очень полезно для создания изображения лица.

Beta-VAE (2)

- β-VAE (Higgins et al., 2017) - модификация VAE с упором на обнаружение распутанных скрытых факторов. Для этого максимизируем вер-ть генерации реальных данных, сохраняя расстояние между реальным и предполагаемым апостериорным распределением небольшим

$$\max_{\phi,\theta} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$$

при ограничении

$$KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) < \delta$$

Beta-VAE (3)

- Ипользуя множитель Лагранжа β, получим

$$L_{\text{BETA}}(\phi, \beta) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + \beta KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

- Когда $\beta = 1$, это VAE.
- Когда $\beta > 1$, получаем более сильное ограничение на скрытое представление и ограничиваем способность представления $\mathbf{z}$.
- Больший β способствует более эффективному скрытому представление и дальнейшему распутыванию.
- Между тем, больший β может привести к компромиссу между качеством реконструкции и степенью распутывания.

VQ-VAE (1)

- Модель VQ-VAE («Vector Quantized-Variational AutoEncoder»; ван ден Оорд и др., 2017) обучает дискретную скрытую переменную с помощью кодера, поскольку дискретные представления могут более естественно подходить для таких задач, как язык, речь, рассуждения, и т.д.
- Векторное квантование (VQ) — это метод отображения K-мерных векторов в конечный набор «кодовых» векторов. Процесс очень похож на алгоритм KNN. Оптимальный кодовый вектор центроида, в который должен быть отображен образец, имеет минимальное евклидово расстояние.

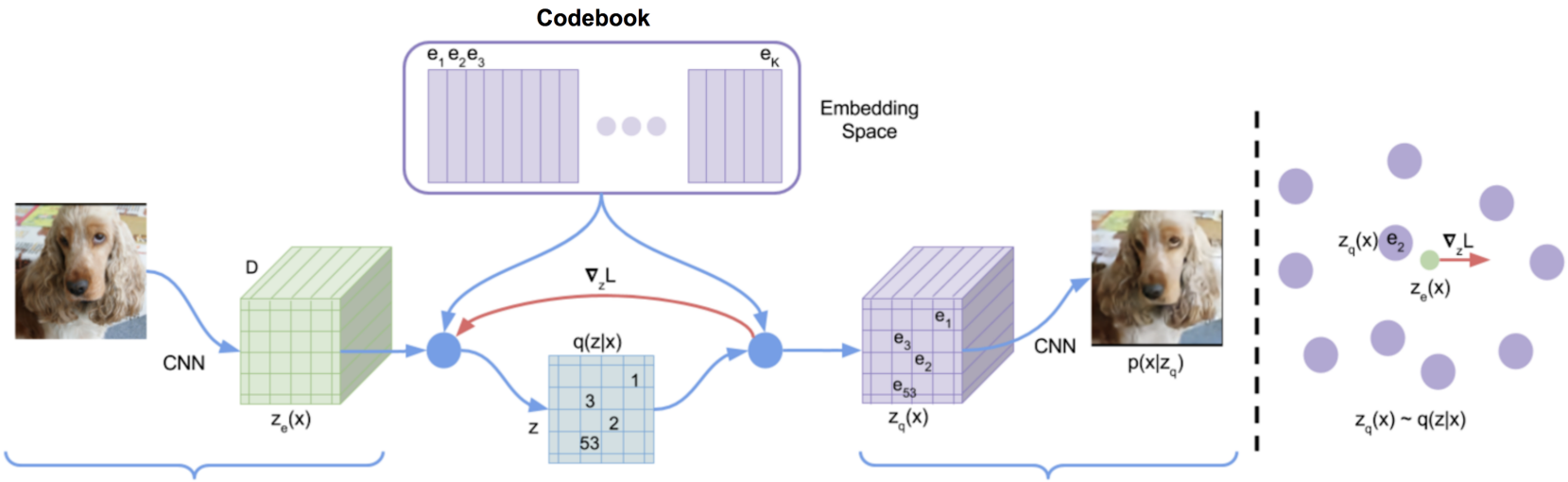
VQ-VAE (2)

- Пусть $\mathbf{e} \in \mathbb{R}^{K \times D}, i = 1, \ldots, K$ - скрытый эмбединг ("кодовая книга") в VQ-VAE, где K - количество категорий скрытых переменных и D - размер эмбединга. Индивидуальный эмбединг - $\mathbf{e}_i \in \mathbb{R}^D, i = 1, \ldots, K$.
- Выход кодировщика $E(\mathbf{x}) = \mathbf{z}_e$ осуществляется через поиск ближайшего соседа, чтобы соответствовать одному из K векторов эмбедингов, а затем совпадающий кодовый вектор становится входом для декодера $D(.)$:

$$\mathbf{z}_q(\mathbf{x}) = \text{Quantize}(E(\mathbf{x})) = \mathbf{e}_k \text{ где } k = \arg\min_i \|E(\mathbf{x}) - \mathbf{e}_i\|_2$$

- Дискретные скрытые переменные могут иметь разную форму, например, 1D для речи, 2D для изображения и 3D для видео.

Архитектура VQ-VAE (1)



Posterior categorical distribution:

$$q(\mathbf{z} = \mathbf{e}_k|\mathbf{x}) = \begin{cases} 1 & \text{if } k = \arg\min_i \|\mathbf{z}_e(\mathbf{x}) - \mathbf{e}_i\|_2 \\ 0 & \text{otherwise.} \end{cases}$$

Архитектура VQ-VAE (2)

- Т.к. argmin() не дифференцируем в дискретном пространстве, градиенты $\nabla_z L$ с входа декодера $\mathbf{z}_q$ копируются на выход кодировщика $\mathbf{z}_e$. Помимо потерь при реконструкции, VQ-VAE также оптимизирует:
  - *VQ-Loss*: L2-ошибка между пространством эмбедингов и выходами кодировщика.
  - *Commitment loss*: мера, позволяющая побудить выходные данные кодировщика оставаться близко к пространству эмбедингов и предотвратить их слишком частые отклонения от одного кодового вектора к другому

Архитектура VQ-VAE (3)

$$L = \underbrace{\|\mathbf{x} - D(\mathbf{e}_k)\|_2^2}_{\text{reconstruction loss}} + \underbrace{\|\text{sg}[E(\mathbf{x})] - \mathbf{e}_k\|_2^2}_{\text{VQ loss}} + \underbrace{\beta\|E(\mathbf{x}) - \text{sg}[\mathbf{e}_k]\|_2^2}_{\text{commitment loss}}$$

где $\text{sg}[\cdot]$ - стоп-градиент оператор

Архитектура VQ-VAE (4)

Эмбеддинги в кодовой книге обновляются через EMA (экспоненциальное скользящее среднее). Дан кодовый вектор $\mathbf{e}_i$. Пусть имеем n_i выходные векторы кодировщика $\{\mathbf{z}_{i,j}\}_{j=1}^{n_i}$, которые квантуются до $\mathbf{e}_i$:

$$N_i^{(t)} = \gamma N_i^{(t-1)} + (1-\gamma) n_i^{(t)}$$

$$\mathbf{m}_i^{(t)} = \gamma \mathbf{m}_i^{(t-1)} + (1-\gamma) \sum_{j=1}^{n_i^{(t)}} \mathbf{z}_{i,j}^{(t)}$$

$$\mathbf{e}_i^{(t)} = \mathbf{m}_i^{(t)} / N_i^{(t)}$$

где t размер батча по времени, N_i и $\mathbf{m}_i$ - накопленное количество векторов и объем соответственно.

VQ-VAE-2 (1)

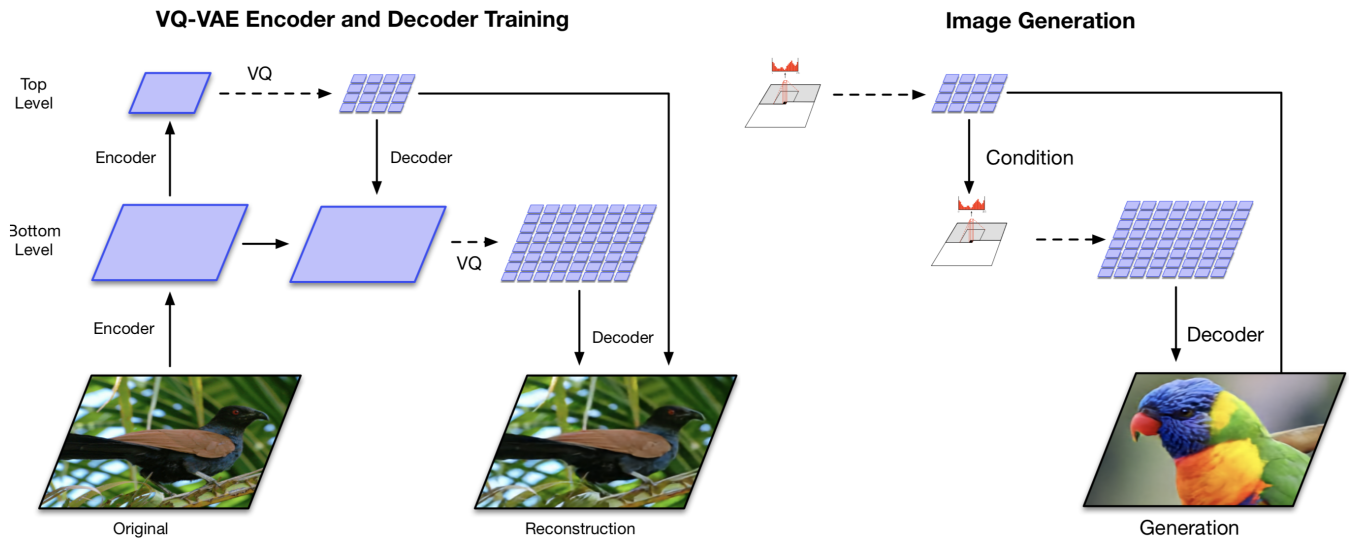
- VQ-VAE-2 (Али Разави и др., 2019) представляет собой двухуровневую иерархическую VQ-VAE в сочетании с авторегрессионной моделью самовнимания.
- **Этап 1** - обучение иерархической модели VQ-VAE: иерархические скрытые переменные предназначен для отделения локальных паттернов (т.е. текстуры) от глобальной информации (т.е. форм объектов). Обучение кодовой книги более нижнего уровня обусловлено меньшим кодом верхнего уровня, поэтому ему не нужно изучать все с нуля.

VQ-VAE-2 (2)

- **Этап 2** - обучение априорного вектора по кодовой книге, чтобы случайно выбирать из нее и генерировать изображения. Т.о. декодер может получать входные векторы, выбранные из того же распределения, что и при обучении. Мощная авторегрессионная модель, дополненная многоуровневыми слоями самовнимания, используется для получения априорного распределения.

VQ-VAE-2 (3)

Учитывая, что VQ-VAE-2 зависит от дискретных скрытых переменных, настроенных в простой иерархиии, качество сгенерированных изображений просто потрясающее



Ресурсы и software

- Variational Autoencoder в TensorFlow: https://jmetzen.github.io/2015-11-27/vae.html
- Demo: http://www.dpkingma.com/sgvb_mnist_demo/demo.html

Вопросы

?